Meeting 2

Summer 2009 Doing DSP Workshop

Today:

- Comments.
- Joining a lab section.
- Introduce FPGAs and VHDL.

The modern computer hovers between the obsolescent and the nonexistent.

— Sydney Brenner, attributed in Science, 5 January 1990.

Workshop CD

- Workshop CD is rather full. Lots of relevant material.
- Most collections have an `xxx.html` (or the like) index file.
- A few don't.
- Poke around. See what's there. The collection is intended to save you time and give some focus.
- There are two DSP oriented text books included.

Comments

- This is not a course. This is a workshop. Less structure, more opportunity to participate, more opportunity to pursue one's own interests.
- Expect to have 6 lab exercises intended to familiarize you with the Workshop hardware. One per week for six weeks.
- Following these exercises you have the opportunity to focus on some aspect of DSP theory and/or implementation.
- The nominal end of the Workshop will be start of August, maybe mid August, well, whenever.
- Discussion sessions Tu and Th 1:00 PM to 2:30 PM.
- The first six weeks of discussions will focus on labs. Some DSP will be included.
- After six weeks will seek volunteers to research a topic and make a minimum 20 minute presentation to the Workshop.

More comments

- We presently have about 28 Workshop members.
- The Workshop web page is at
  www.eecs.umich.edu/courses/doing_dsp/
- A set of interesting questions can be found on the Workshop web site. We will discuss them starting one week from to day. This is NOT a homework assignment. However, you might want to treat it as such.
- There are two lab sections, one Tuesday and one Thursday. The max capacity is 18 students per section. This assumes lab partners. Feel free to use off time.
- The first lab session will be next Tuesday.
- I will be out of town Friday through Monday.

First lab

Introduces the Spartan-3 Starter Board and VHDL.

- Have to start somewhere, some time.
- Read (not study) the *Spartan-3 Starter Kit Board User Guide*. Can be found on the Workshop CD and on the Workshop handouts web page.
- Find a VHDL tutorial and read (not study) it. I've put two onto the handouts web page.
- Chih-Wei will be posting the first week lab write-up probably on Monday. Probably on the handouts web page or distributed by mailing list.
- There will help in lab. No deadline and no required lab report. Though you might might want to sketch one out for future reference.

A common lab exercise theme

Note that the MSP430 and the Piccolo possess at least one channel of A/D conversion. Neither includes a D/A converter.

Kind of need to have a D/A converter if we are going to explore computer generation of analog waveforms, e.g., sine wave.

- Will explore one-bit D/A conversion.
- Two methods, pulse width modulation and sigma-delta.
- What happen once we (re)enter the analog world will be of interest as well.
- Will cheat with the S3SB and use a PMod D/A, at least at first.
- Will be typically in the second week of each exercise pair.
- New to the coordinators as well.

Workshop mailing list

I've included everyone that I'm aware of. If you have not received an email from me in the last two days you can add yourself to the list.

For automatic subscriptions, send email to

`doing_dsp-request@eecs.umich.edu`

with the subject as "Subscribe" or "Unsubscribe".

Note that you can remove yourself as well.

What defines a DSP microcomputer?

For years I've defined DSP microcomputer as a microcomputer that possesses an instruction that multiplies two numbers together and adds the result into an accumulator. A MAC instruction.

Where does this leave microcomputers like the MSP430F2012? It doesn't even have a multiplier. However, it does have a 10-bit A/D converter. It obviously is meant to do some sort of digital signal processing.

Revised definition:

> *A DSP microcomputer is one whose instruction set includes a multiply-and-accumulate (MAC) instruction and/or possesses an A/D converter.*

References

The best introduction to VHDL that I've seen is:

Circuit Design with VHDL, V. Pedroni ($29–$42).

The library copy appears to have been mis-shelved. I've put my and Professor Stark's copies on reserve.

Another good, reasonably priced book, is *Advanced Digital Logic Design* by Sunggu Lee.

See the doing_dsp handouts web page for:

- *ISE 10.1i Quick Start Tutorial*.
- *Programmable Logic Design Quick Start Handbook*.
- *Spartan-3 Starter Board User's Guide*.
- Chih-Wei's *ISE Tutorial*.

What is an FPGA?

Field **P**rogrammable **G**ate **A**rray

A programmable logic device that has lots of gates and other special features that allow its “easy” use in digital systems. Typically volatile, needs a boot device.

Xilinx introduced first FPGA devices in 1985. Best guess is that there have been 6 or 7 generations. Current technology, Virtex-5, is at the 65 nm level. The Spartan-3 devices used in the lab are 90 nm.

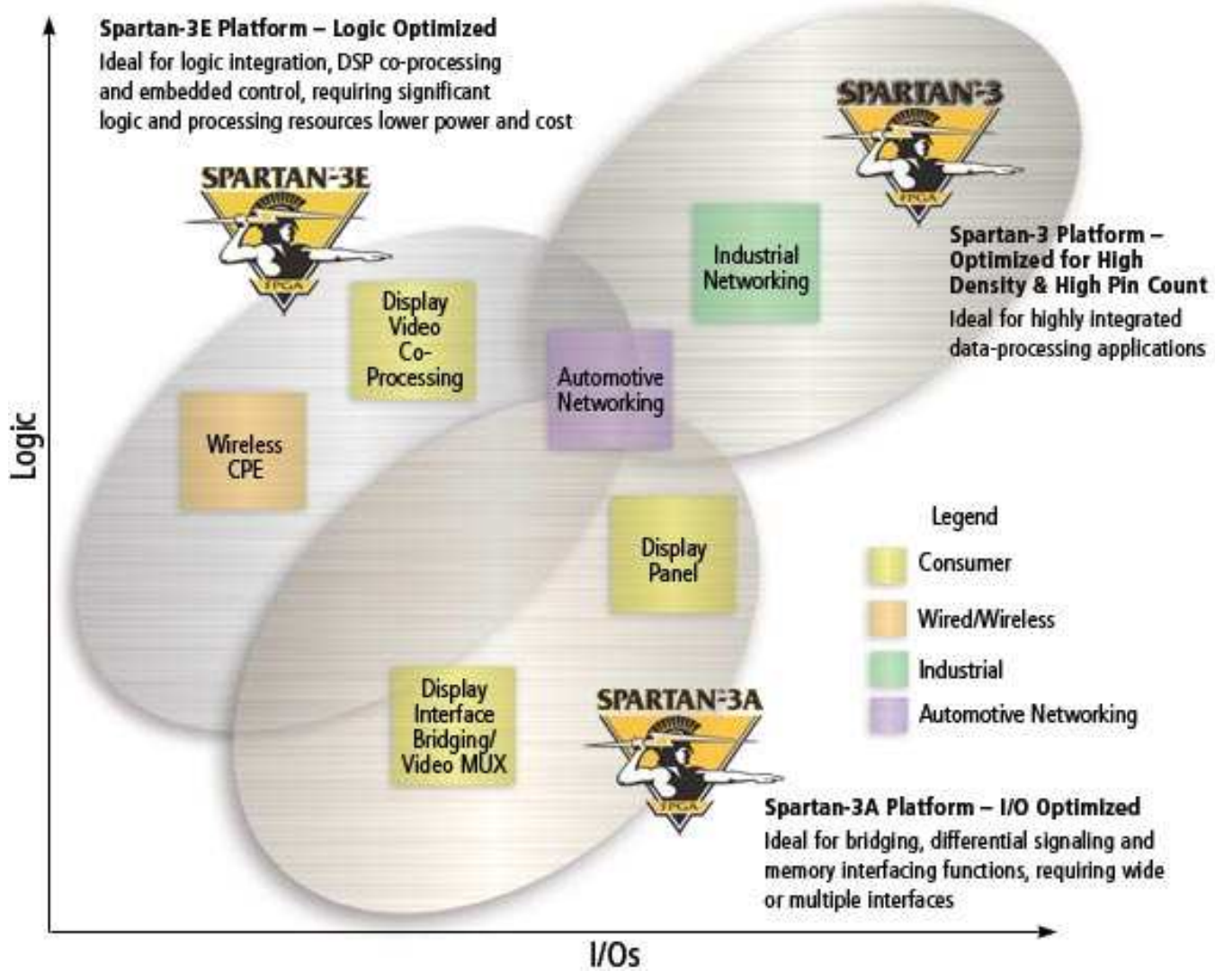
Currently Xilinx has greater than 50% market share. 2008 net revenue was $1.9 billion.

Costs are dropping, capabilities are rising.

Where FPGAs find use?

- Low volume (< 15k) applications.
- Where reconfigurability is important.
- Augmenting DSP chips.
- Replacing DSP chips.
- Software defined radios.
- Broadband modems.
- Set-top boxes.
- Auto-entertainment.
- Industrial automation.
- Many, many other applications.

The Xilinx Spartan-3 family devices


Spartan-3E Platform – Logic Optimized
Ideal for logic integration, DSP co-processing and embedded control, requiring significant logic and processing resources lower power and cost
SPARTAN-3E
SPARTAN-3
Spartan-3 Platform – Optimized for High Density & High Pin Count
Ideal for highly integrated data-processing applications
Industrial Networking
Display Video Co-Processing
Automotive Networking
Wireless CPE
Display Panel
Display Interface Bridging/ Video MUX
SPARTAN-3A
Spartan-3A Platform – I/O Optimized
Ideal for bridging, differential signaling and memory interfacing functions, requiring wide or multiple interfaces
Legend
Consumer
Wired/Wireless
Industrial
Automotive Networking
Logic
I/Os

From the Xilinx web site.

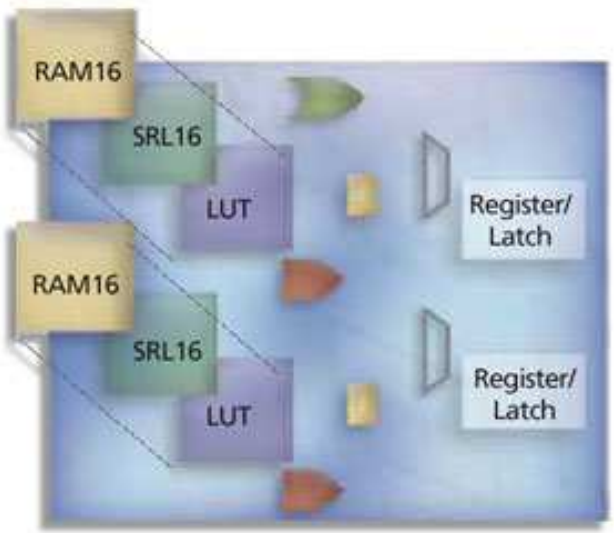
Spartan-3 features

- SelectIO™ signaling
  - Up to 784 I/O pins
  - 622 Mb/s data transfer rate per I/O
  - 18 single-ended signal standards
  - 6 differential I/O standards including LVDS, RSDS
  - Termination by Digitally Controlled Impedance
  - Signal swing ranging from 1.14V to 3.45V
  - Double Data Rate (DDR) support
- Logic resources
  - Abundant logic cells with shift register capability
  - Wide multiplexers
  - Fast look-ahead carry logic
  - Dedicated 18 x 18 multipliers
  - JTAG logic compatible with IEEE 1149.1/1532
- SelectRAM™ hierarchical memory
  - Up to 1,872 Kbits of total block RAM
  - Up to 520 Kbits of total distributed RAM
- Digital Clock Manager (up to four DCMs)
  - Clock skew elimination
  - Frequency synthesis
  - High resolution phase shifting
- Eight global clock lines and abundant routing

From Xilinx Spartan-3 data manual

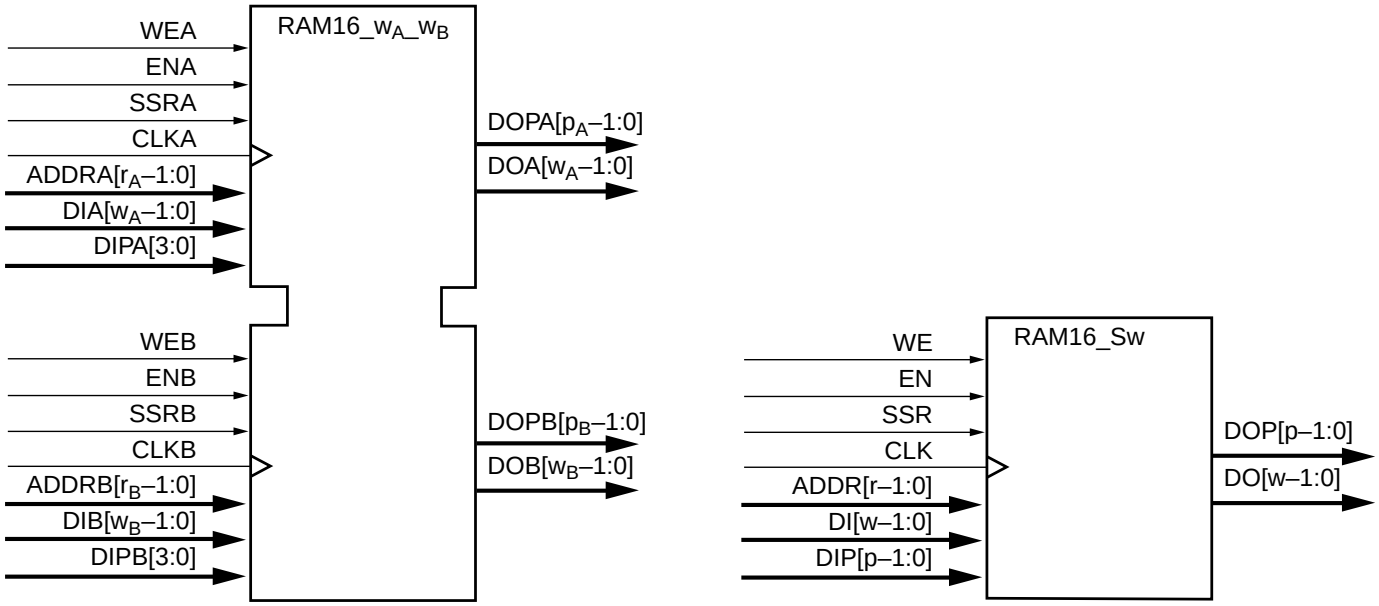
Configurable Logic Block

- CLB is optimized for area and speed for compact high performance design.
- Four slices per CLB implement any combinatorial and sequential circuit.
- Each slice has 4-input look-up tables (LUT), flip-flops, multiplexors, arithmetic logic, carry logic, and dedicated internal routing.
- Dedicated AND/OR logic implements wide input functions.



Xilinx web site.

Off fabric 18K bit memory blocks



S3E-100	4 blocks	72,728 bits
S3-200	12 blocks	221,184 bits
S3-1000	24 blocks	442,368 bits

From Xilinx application note: XAPP463 V2.0.

Block RAM configurations

Total RAM bits, including parity	18,432 (16K data + 2K parity)
Memory Organizations	16Kx1 8Kx2 4Kx4 2Kx8 (no parity) 2Kx9 (x8 + parity) 1Kx16 (no parity) 1Kx18 (x16 + 2 parity) 512x32 (no parity) 512x36 (x32 + 4 parity) 256x72 (single-port only)
Parity	Available and optional only for organizations greater than byte-wide. Parity bits optionally available as extra data bits.
Performance	200 MHz (estimated)
Timing Interface	Simple synchronous interface. Similar to reading and writing from a register with a setup time for write operations and clock-to-output delay for read operations.
Single-Port	Yes
True Dual-Port	Yes
ROM, Initial RAM Contents	Yes
Mixed Data Port Widths	Yes
Power-Up Condition	User-defined data, defaults to zero

From Xilinx application note: XAPP463 V2.0.

Off fabric multipliers

Data Flow

Each embedded multiplier block (MULT18X18 primitive) supports two independent dynamic data input ports: 18-bit signed or 17-bit unsigned. The two inputs are referred to as the multiplicand and the multiplier, or the factors, while the output is the product. The MULT18X18 primitive is illustrated in Figure 1.

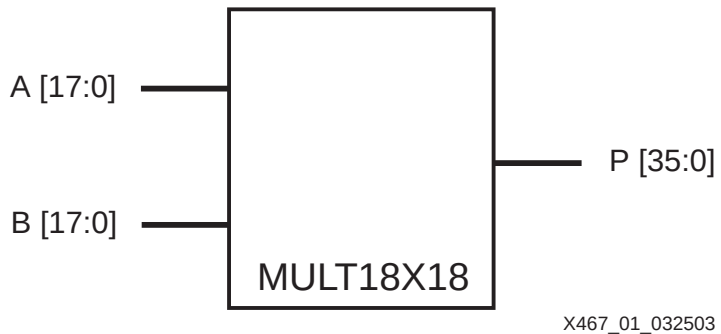


Figure 1: **Embedded Multiplier**

Can be configured as combinational or as registered. Timing depends on the number of (lower) bits used in result. Note: 17-bit unsigned or 18-bit two's complement.

From Xilinx application note: XAPP467.

Multiplier timing–using Core Generator

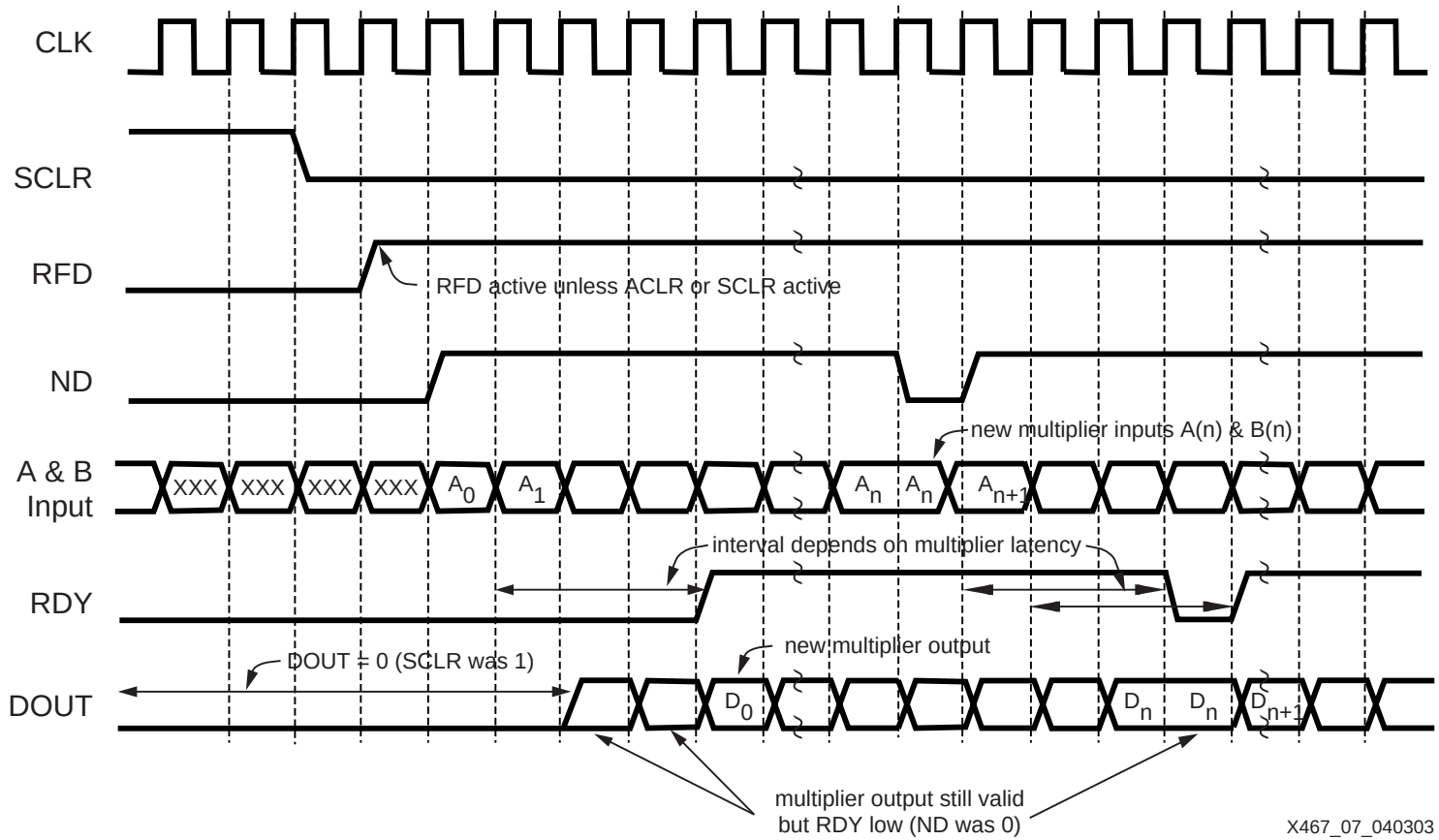
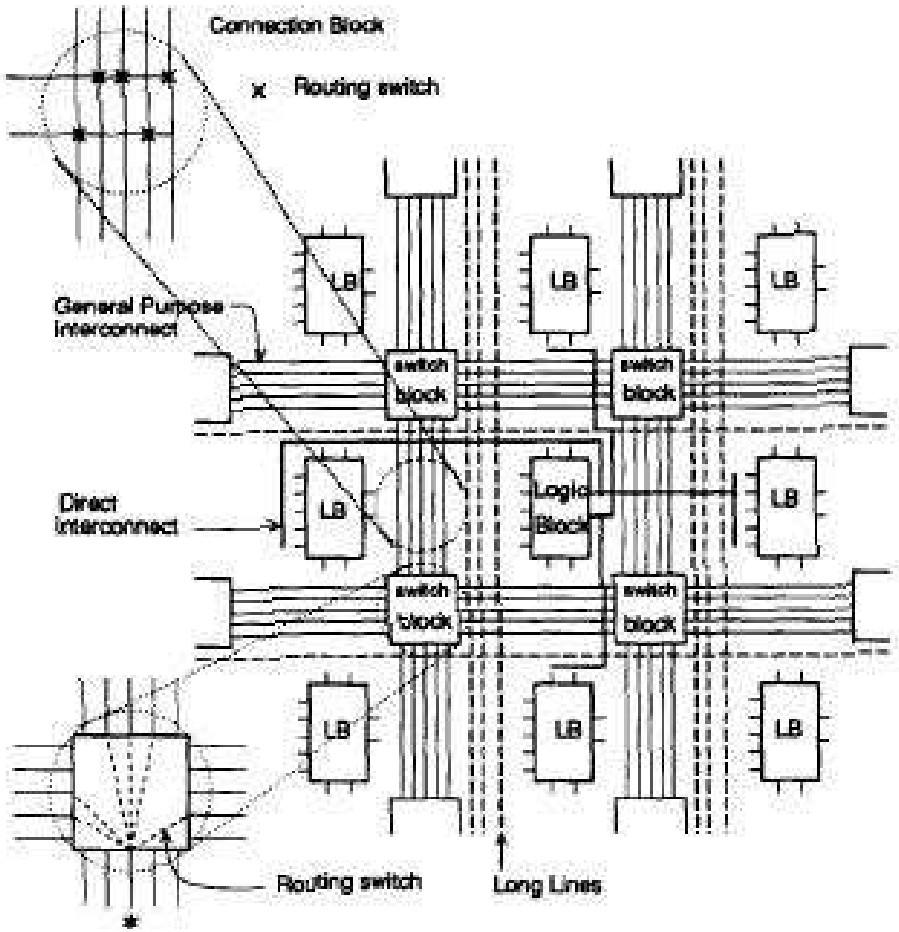


Figure 9: **Multiplier Generator Timing Diagram**

From Xilinx application note: XAPP467.

Routing–what makes it all happen



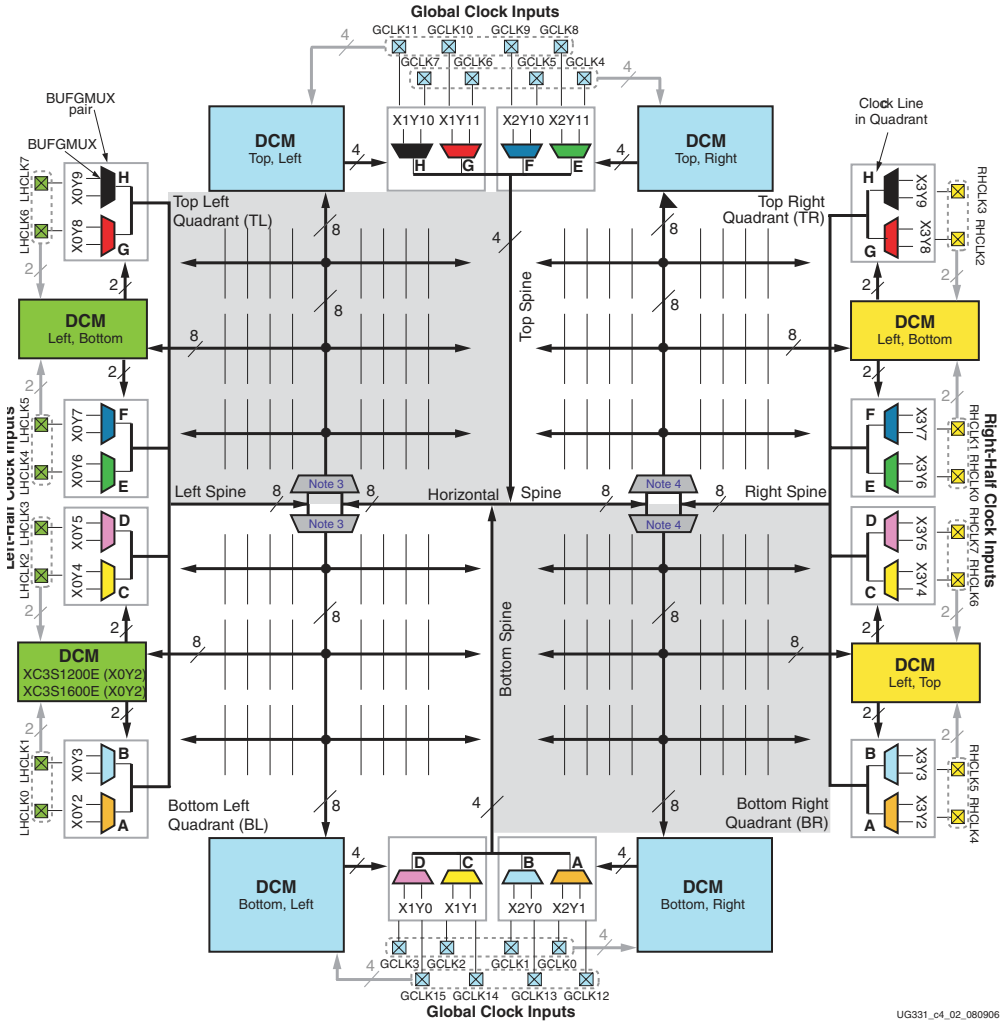
Xilinx routing architecture.

There are four types of wire segments available :

- general purpose segments, the ones that pass through switches in the switch block.
- Direct interconnect : ones which connect logic block pins to four surrounding connecting blocks
- long line : high fan out uniform delay connections
- clock lines : clock signal provider which runs all over the chip.

From http://www.tutorial-reports.com/computer-science/fpga/routing.php?PHPSESSID=9ed5dd1540f5b90fc4088f7c14

and when it happens



From Xilinx UG331.pdf, Spartan-3 Generation FPGA User Guide (preliminary).

Digital Clock Multipliers

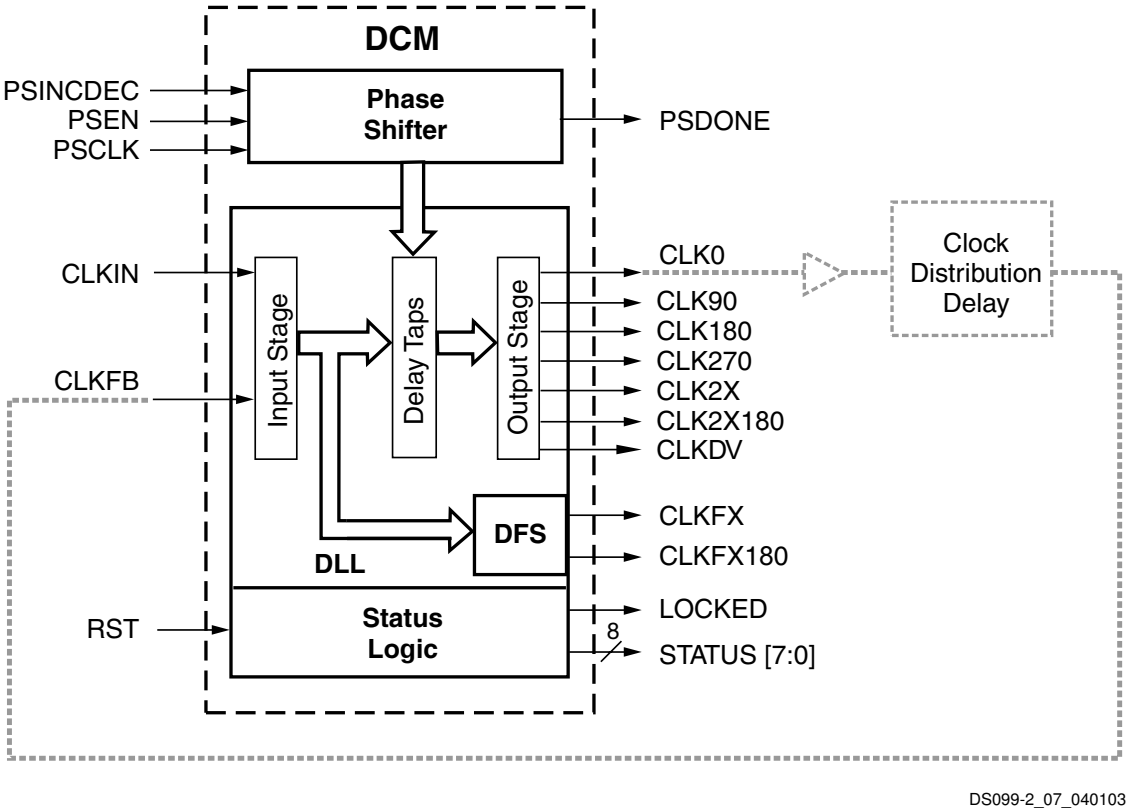


Figure 3-3: **DCM Functional Block Diagram**

From Xilinx UG331.pdf, Spartan-3 Generation FPGA User Guide (preliminary).

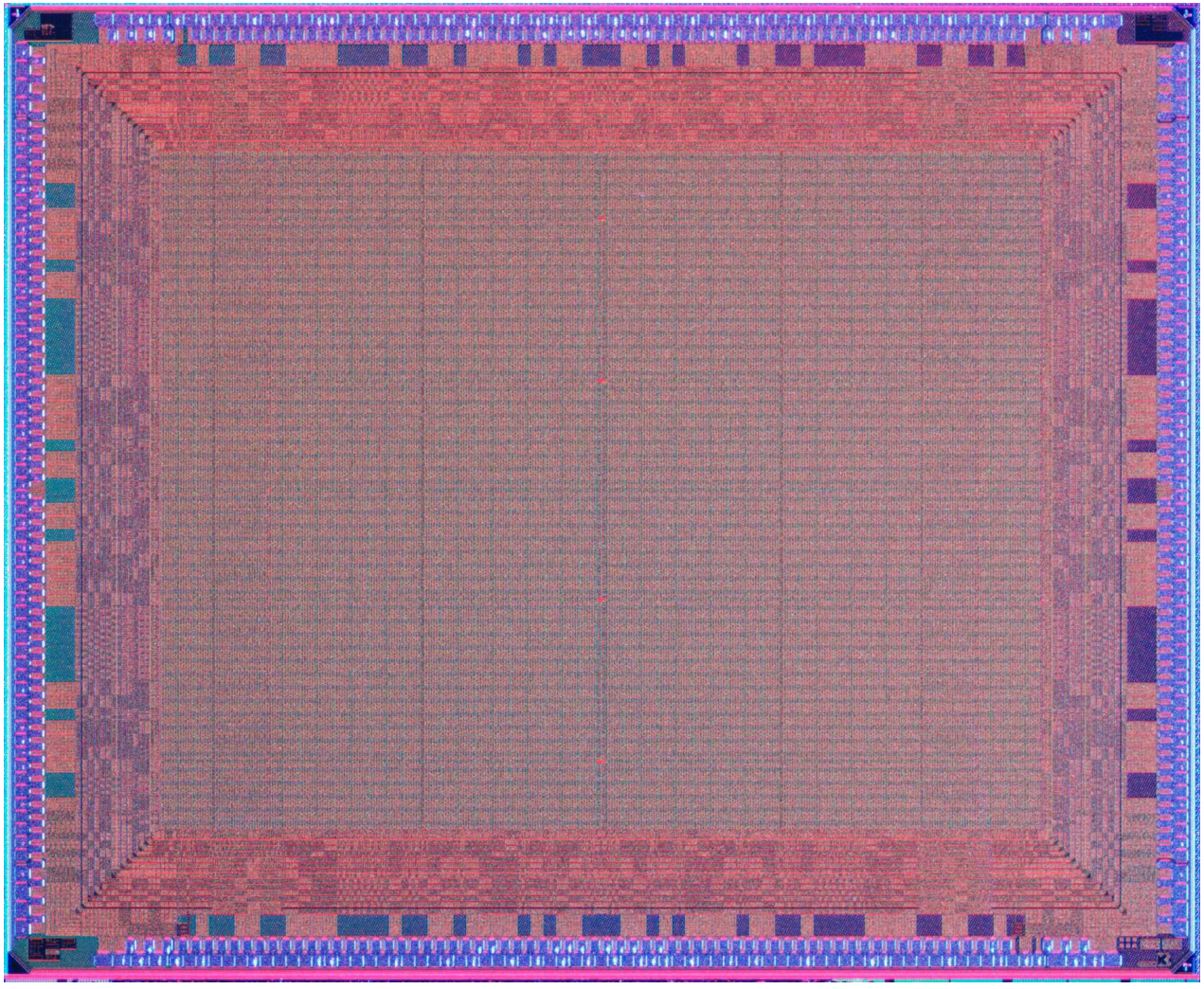
IO “standards”

“The nice things about standards is that there are so many of them”.

S3	24 differential and single ended
S3E	18 differential and single ended
S3A	26 differential and single ended

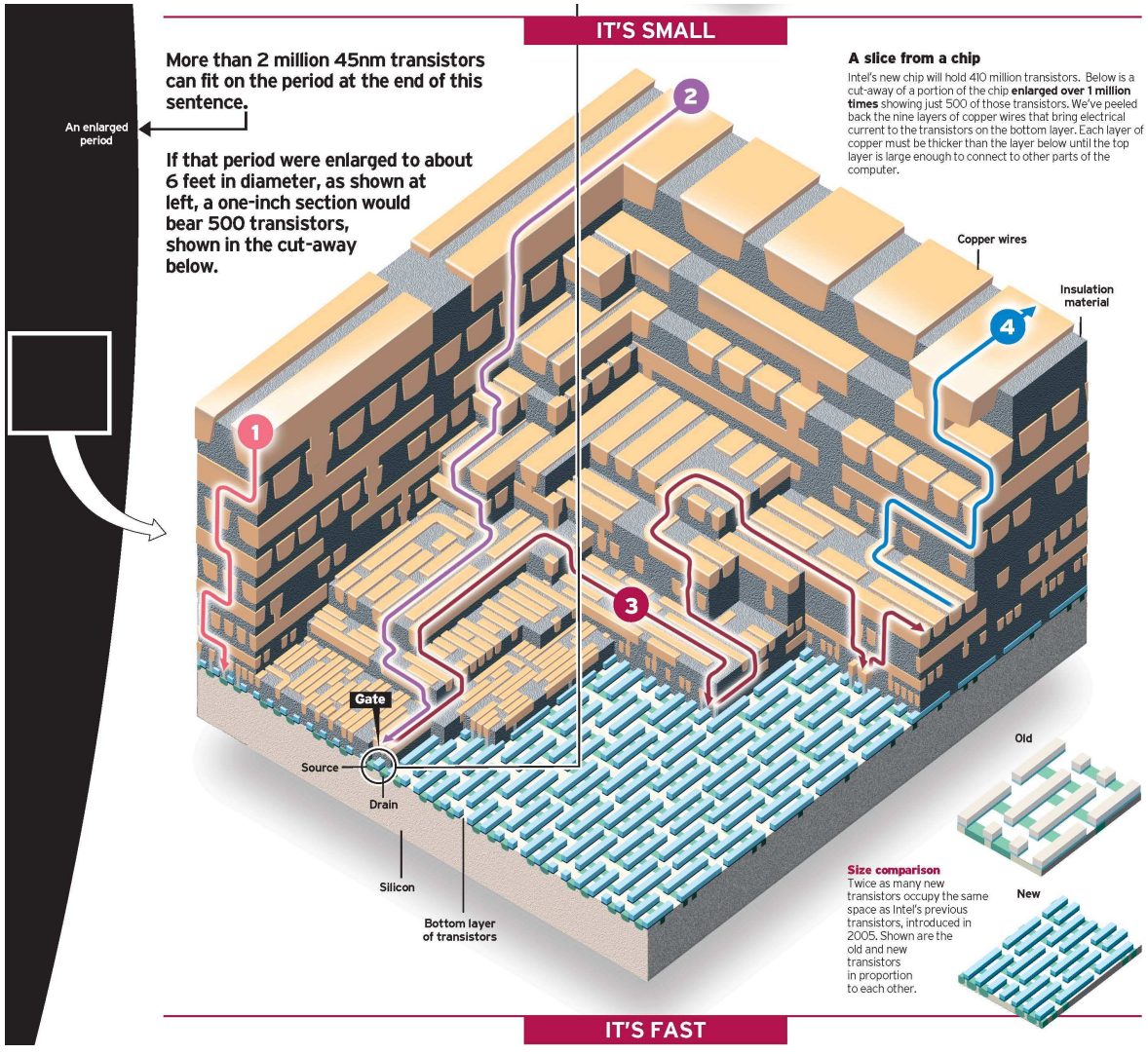
Some restrictions apply.

The Spartan-3 die

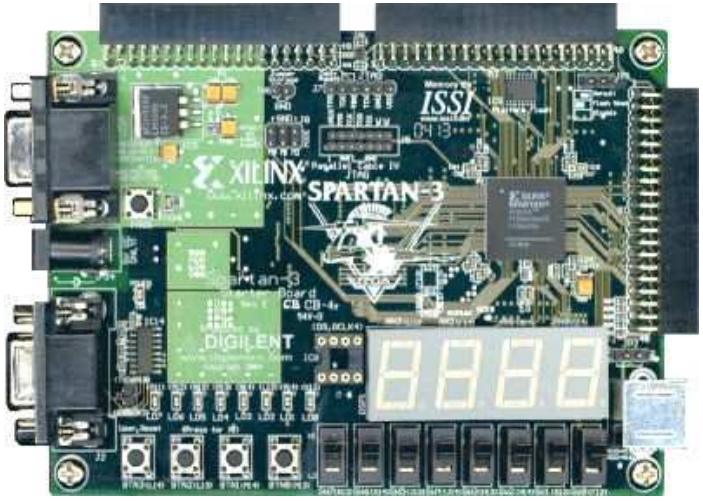


From Xilinx

An INTEL figure


IT'S SMALL
More than 2 million 45nm transistors can fit on the period at the end of this sentence.
An enlarged period
If that period were enlarged to about 6 feet in diameter, as shown at left, a one-inch section would bear 500 transistors, shown in the cut-away below.
A slice from a chip
Intel's new chip will hold 410 million transistors. Below is a cut-away of a portion of the chip enlarged over 1 million times showing just 500 of those transistors. We've peeled back the nine layers of copper wires that bring electrical current to the transistors on the bottom layer. Each layer of copper must be thicker than the layer below until the top layer is large enough to connect to other parts of the computer.
Copper wires
Insulation material
1
2
3
4
Gate
Source
Drain
Silicon
Bottom layer of transistors
Old
New
Size comparison
Twice as many new transistors occupy the same space as Intel's previous transistors, introduced in 2005. Shown are the old and new transistors in proportion to each other.
IT'S FAST

The Spartan-3 starter board



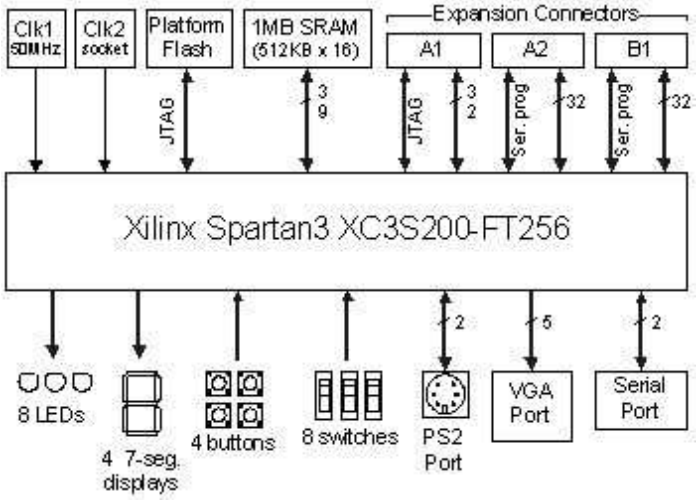


Photo and diagram from the Digilent web site.

Basic unit (200K gates) sold by Xilinx for $99.

Manufactured by Digilent. Sells boards using 200K, 400K or 1M gate Spartan-3 for up to $149. Also sells auxiliary boards such as ethernet interface, 1 MHz dual A/D and D/A, etc.

Well supported by Xilinx's *free* WebPack version of its ISE 8.2 development software.

Uses 50 MHz clock (can use on-chip clock multiplier to increase).

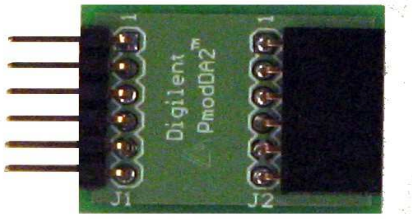
FPGA boards in play in and about

EECS 452 is presently focused on Xilinx Spartan-3 FPGA devices and is using starter/evaluation boards produced by Digilent.

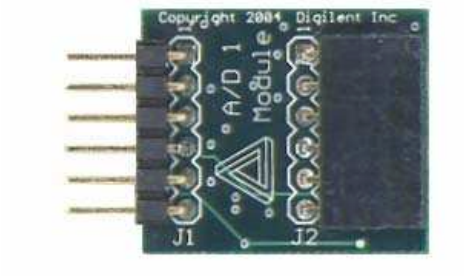
- Spartan-3 1000 Starter Board. Used in lab.
- Spartan-3 200 Starter Board.
- Spartan-3 1000 Nexys Board.
- Spartan-3E 500 Nexys-2 Board.
- Spartan-3E 100 Basys board (2). .
- Virtex-2 Pro XUP. Available for project use. Xilinx donated.
- Nallatech uses Virtex-II 2V3000-4. Donated.
- Altera DE2 (as used in EECS 270).
- Spartan-3E 500 Starter Kit. .

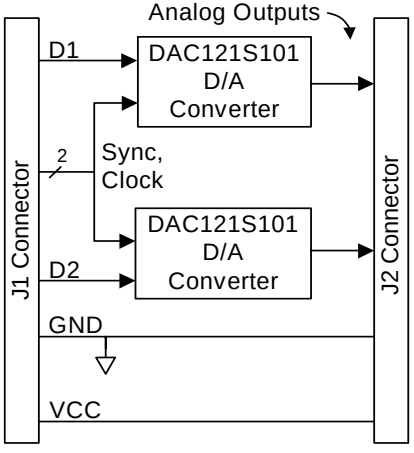
These boards are reasonably “compatible” but not necessarily without a bit of care and effort.

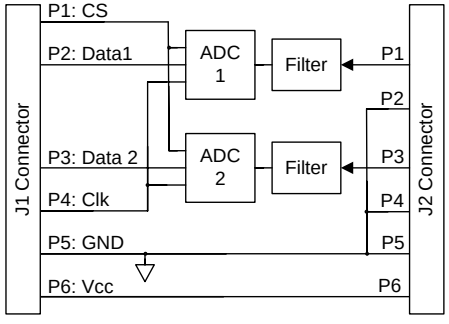
Digilent D/A, A/D and MIB



Digilent PmodDA2







AD1 Circuit Diagram

From Digilent data sheets.

Pmod in and out

- Pins on MIB connect to pins on PMOD modules.
- Use socket-socket cables to connect.
- Make sure VB on MIB connects to VCC on PMOD!
- Make connections with power OFF!
- Use 5 pin header on PMOD output so power is not present!

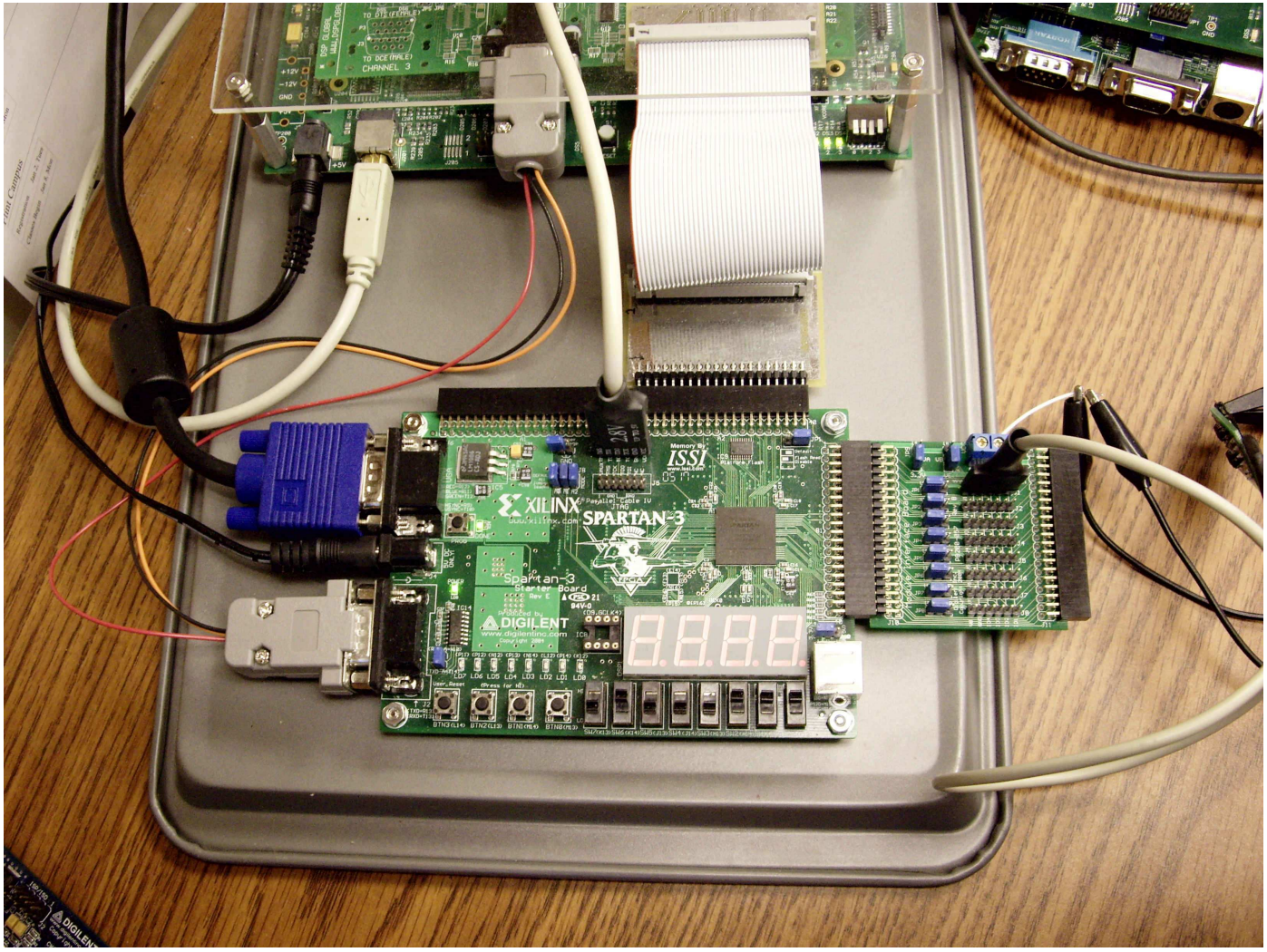
For now connector A1 is “not” being used. (Actually is connected to memory bus which we will be using. Avoid using.

A2 is reserved for connecting to DSK peripheral interface bus.

B1 is left for use by other devices. EECS 452 attaches a MIB to B1.

The Spartan-3 Starter Boards has provision for lots of connections. This is the primary reason we chose it.

Lab arrangement



Xilinx tools

- ISE (we use the WebPACK version)
  - VHDL/Verilog/Schematic compiler
  - place and route
  - maps net list into FPGA
  - estimates timing
- Impact
  - programs FPGA and/or EPROM.
  - has support for JTAG programmers.
- ModelSim (by Mentor Graphics)
  - simulates design

Digilent also sells programming cables and has associated free programming software, ExPort.

Lab 1: VHDL, ISE and the S-3 Starter Board

- Demonstrates use of the:
  - Xilinx's ISE and Impact tools,
  - VHDL,
  - slide switches,
  - push buttons (not physically debounced),
  - seven-segment displays,
  - individual LEDs,
  - serial RS-232 interface,
  - VGA output.

Be careful! Don't let the smoke out!

Hardware Description Languages

Originally developed to simulate designs. However, you probably should build what you simulate so they also became implementation languages as well.

Verilog — used in Engin 100, EECS 270 and . . .

- Simple data types.
- Relatively easier to learn.
- Will guess at what you meant.

VHDL — used in EECS 452

- Strongly typed.
- Less easily learned.
- Makes you say what you mean and mean what you say.

“Synthesizable” means that a description can be made into hardware.

Why use an HDL?

- allows modularization, encapsulation, information hiding.
- parameters essentially map into physical connections.
- checks for errors and inconsistencies.
- enhances designer productivity (really!).
- working at a level closer to the device is probably overwhelming.

VHDL supports functional and structural models.

What is Xilinx WebPACK ISE?

In lab we are using Xilinx's WebPACK ISE 10.1 with SP3.

Free subset of ISE Foundation (very good and sort of expensive).

An integrated system design environment.

- Design entry – schematic, VHDL, Verilog.
- Generation of FPGA bit file.
- Simulation support.
- Fits and routes design.
- Includes Impact. . . device programmer.
  - Converts bit to various files.
  - Loads programming into devices.
  - Programs.
- And much, much, more!

FPGA computers from Xilinx

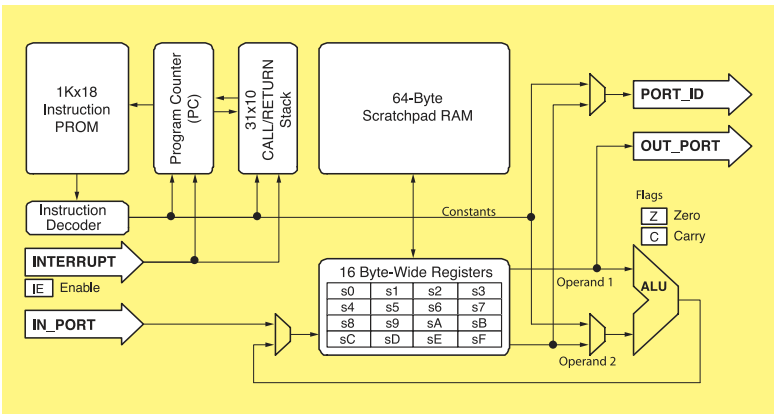
- PicoBlaze — 8-bit, tiny.
  Distributed as VHDL. Very useful. Good starting point.
- MicroBlaze — 32-bit, Unix able, on-fabric.
  Need EDX (he have). Distributed as net list? Pretty much full featured.
- PowerPC — 32-bit, off fabric, integer.
  Need EDK (we have). Two projects have used. FPU available as add on.

PicoBlaze System

Key Feature Set*

- 16 byte-wide general-purpose data registers
- 1K instructions of programmable on-chip program store, automatically loaded during FPGA configuration
- Byte-wide Arithmetic Logic Unit (ALU) with CARRY and ZERO indicator flags
- 64-byte internal scratchpad RAM
- 256 input and 256 output ports for easy expansion and enhancement
- Automatic 31-location CALL/RETURN stack
- Predictable performance, always two clock cycles per instruction, up to 200 MHz or 100 MIPS in a Virtex-4™ FPGA and 88 MHz or 44 MIPS in a Spartan-3 FPGA
- Fast interrupt response; worst-case 5 clock cycles
- Assembler, instruction-set simulator support

PicoBlaze Block Diagram*



PicoBlaze Instruction Set*

Program Control

JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa

CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL C,aaa
CALL NC,aaa

RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC

Input/Output

INPUT sX,pp
INPUT sX,(sY)
OUTPUT sX,pp
OUTPUT sX,(sY)

Logical

LOAD sX,kk
AND sX,kk
OR sX,kk
XOR sX,kk
TEST sX,kk
LOAD sX,sY
AND sX,sY
OR sX,sY
XOR sX,sY
TEST sX,sY

Shift and Rotate

SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX
SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX

Arithmetic

ADD sX,kk
ADDCY sX,kk
SUB sX,kk
SUBCY sX,kk
COMPARE sX,kk
ADD sX,sY
ADDCY sX,sY
SUB sX,sY
SUBCY sX,sY
COMPARE sX,sY

Storage

FETCH sX,ss
FETCH sX,(sY)
STORE sX,ss
STORE sX,(sY)

Interrupt

RETURNI ENABLE
RETURNI DISABLE
ENABLE INTERRUPT
DISABLE INTERRUPT

All instructions execute in 2 clock cycles

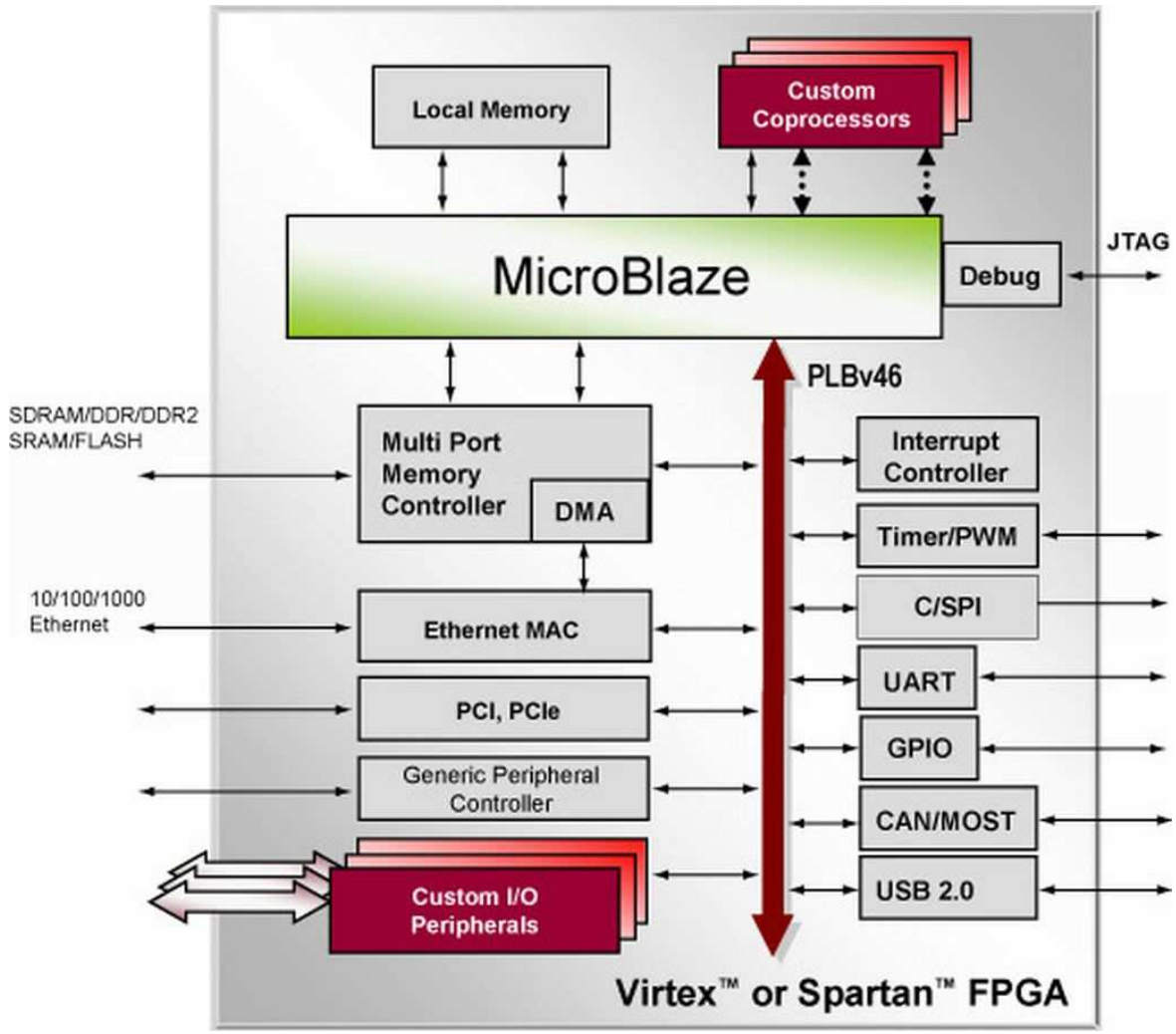
Take the Next Step

Visit **www.xilinx.com/picoblaze** to download the free PicoBlaze microcontroller reference design, which includes the PicoBlaze VHDL source code, assembler, and related documentation.

**Based on PicoBlaze for Spartan-3,Virtex-II/Pro and Virtex-4 (KCPSM3).*

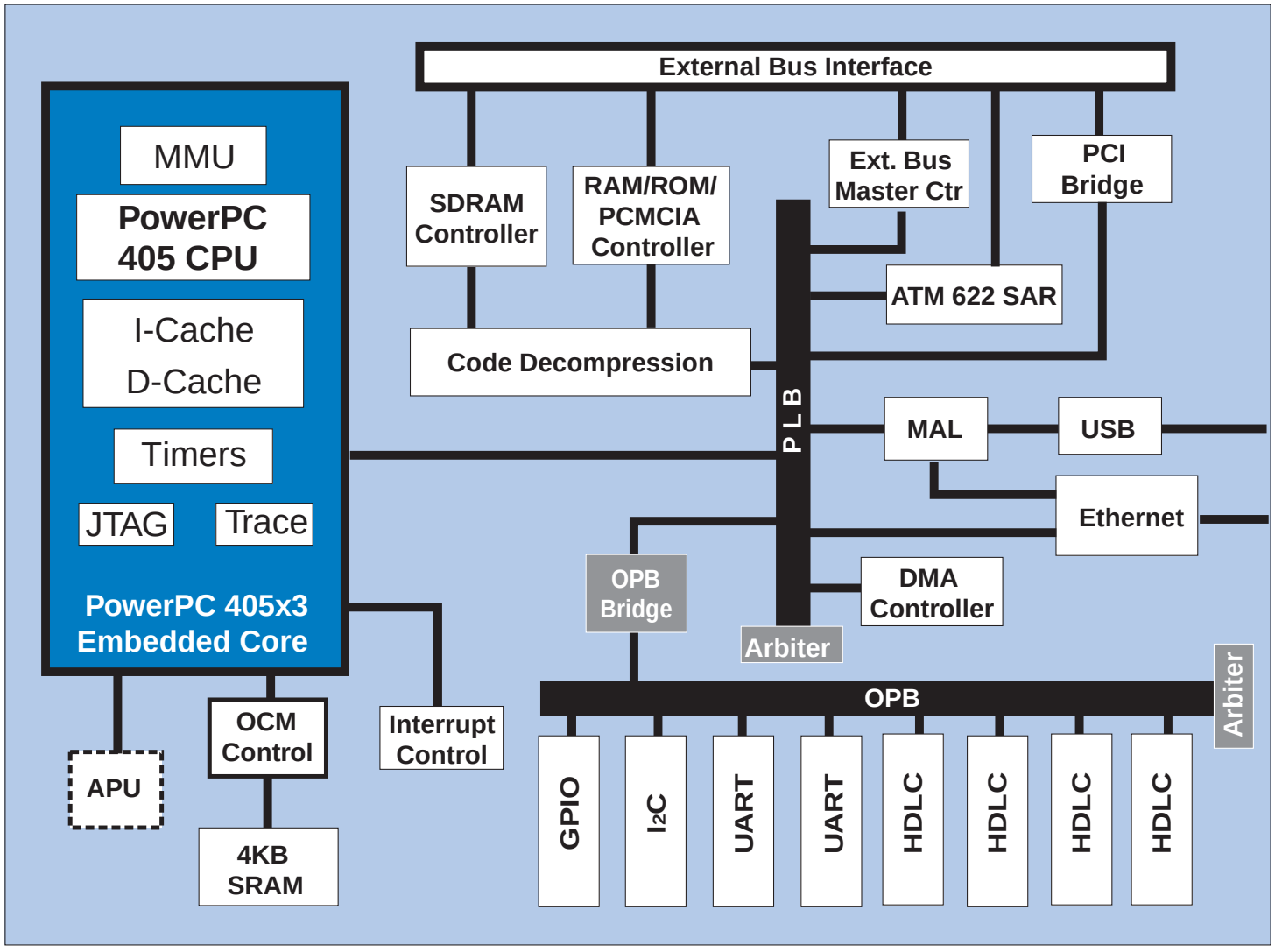
From Xilinx brochure.

MicroBlaze System



From Xilinx brochure.

IBM PowerPC

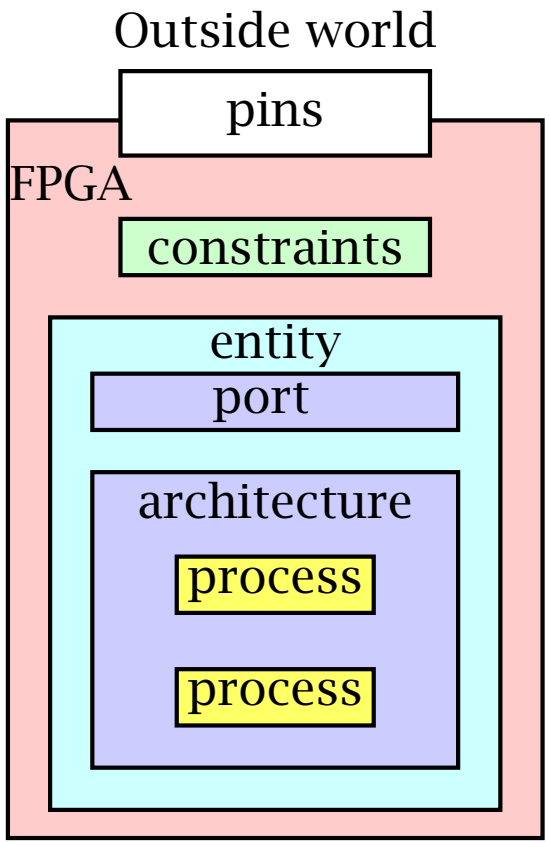


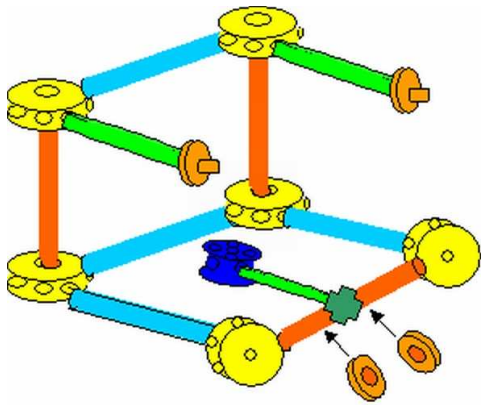
From IBM brochure.

VHDL background

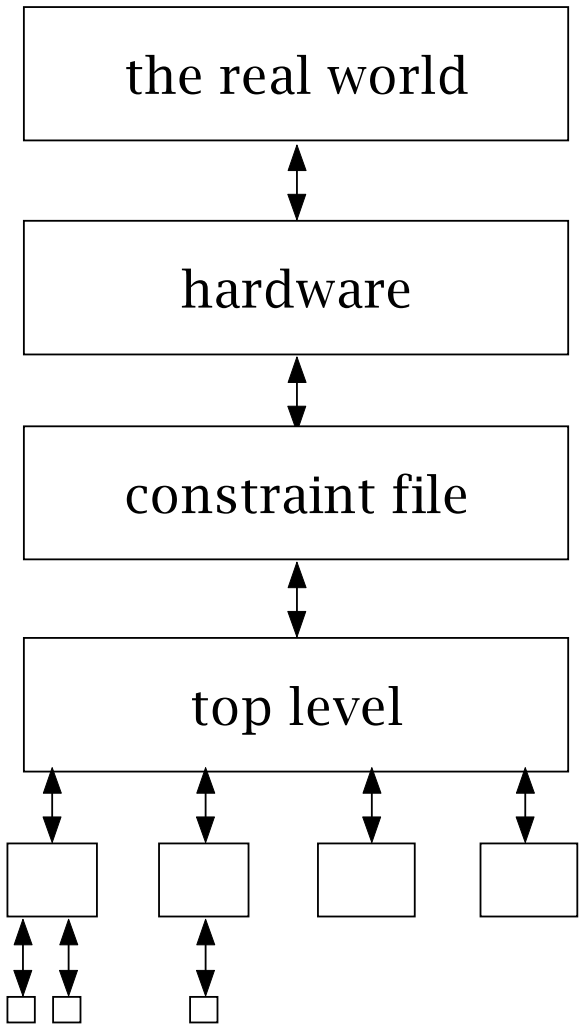
- Hardware design/simulation language.
- Supports structural and behavior design description.
- Logic units are called *entities*.
  - port description... visible connections,
  - architecture... the design, internal signal definitions.
- DoD initiated... ADA like.
- Strongly typed.
- Picky, picky, picky.
- Mostly used in Universities, DoD, East Coast, Europe.

Entities are connectable logic blocks





The design description hierarchy



doing something useful

FPGA and whatever it is connected to

maps signals to FPGA pins

connects major design units

decreasing level of abstraction

nitty-gritty

Example: binary addition

Consider adding two 8-bit values a and b

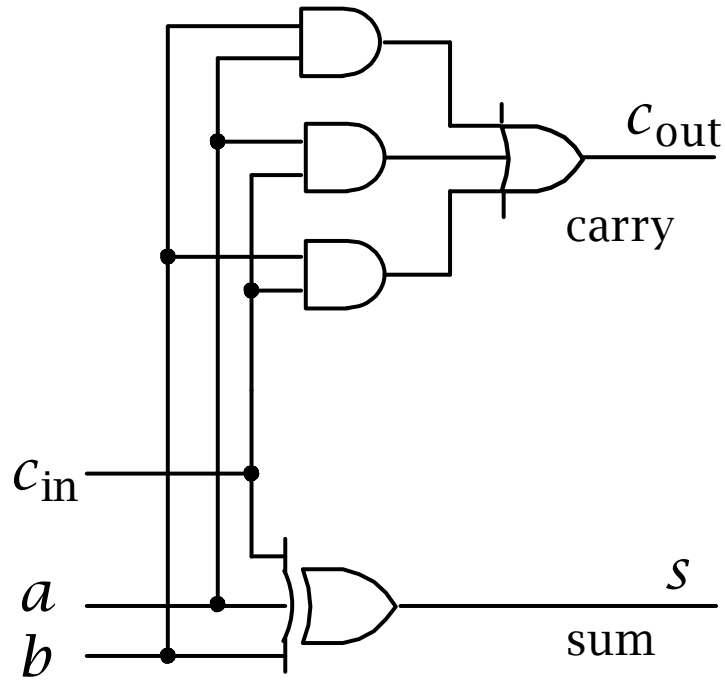
$$\begin{array}{cccccccccc} & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ + & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0. \\ \hline \end{array}$$

For each bit position we form the sum of the two bits in that position and add in any carry from the previous (lower index) bit position. The carry into position 1 is 0.

$$\begin{array}{cccccccccc} & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 & 0 \\ & & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ + & & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ \hline & c_7 & s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 \end{array}$$

A single bit adder unit

outputs		inputs		
c_n	s_n	b_n	a_n	c_{n-1}
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
1	0	1	0	1
1	0	1	1	0
1	1	1	1	1



VHDL – behavioral

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FullAdder01 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           sum : out  STD_LOGIC;
           cout : out  STD_LOGIC);
end FullAdder01;

architecture Behavioral of FullAdder01 is

begin

   sum <= a xor b xor cin;
   cout <= (a and b) or (cin and a) or (cin and b);

end Behavioral;
```

Basic VHDL entity

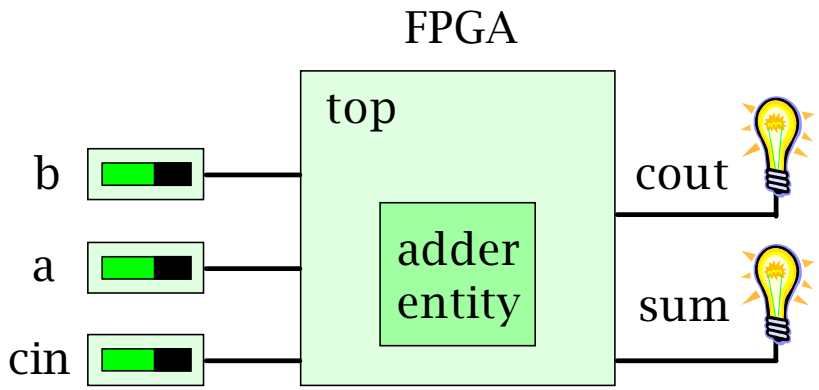
- This defines a “black box”, entity.
- There is a *port* into/outof the box/entity.
  - There are three input signals: a, b, c_in.
  - There are two output signals: sum, c_out.
- Externally the contents of the box are hidded.
- In effect it simply works.
- Associated with the entity port is a description of how it works, the entity’s *architecture*.
- The architecture
  - defines the internal signals.
  - describes what the entity does in order to go from input to output.
  - For this example, everything happens all at once.

An application

Classroom one-bit adder demonstration.

Three slide switches for input. Two LEDs for display.

Allows student to compare logic design operation against truth table.



Top level

- Usually use a top level to connect various module at the top level.
- Normally will not do any logic at this level. Only connect modules.
- Top's port connects to the FPGA I/O pins.
- A xxxx.ucf file is used to attach names to the FPGA pins.
- Pins are attached to external hardware by printed circuit.

```
entity AdderDemoTop is
    Port ( swt : in  STD_LOGIC_VECTOR (2 downto 0);
           led : out  STD_LOGIC_VECTOR (1 downto 0));
end AdderDemoTop;

architecture Behavioral of AdderDemoTop is

begin

    onebitadd : entity work.FullAdder01
    port map( a => swt(1), b => swt(2), cin => swt(0),
              sum => led(0), cout => led(1));

end Behavioral;
```

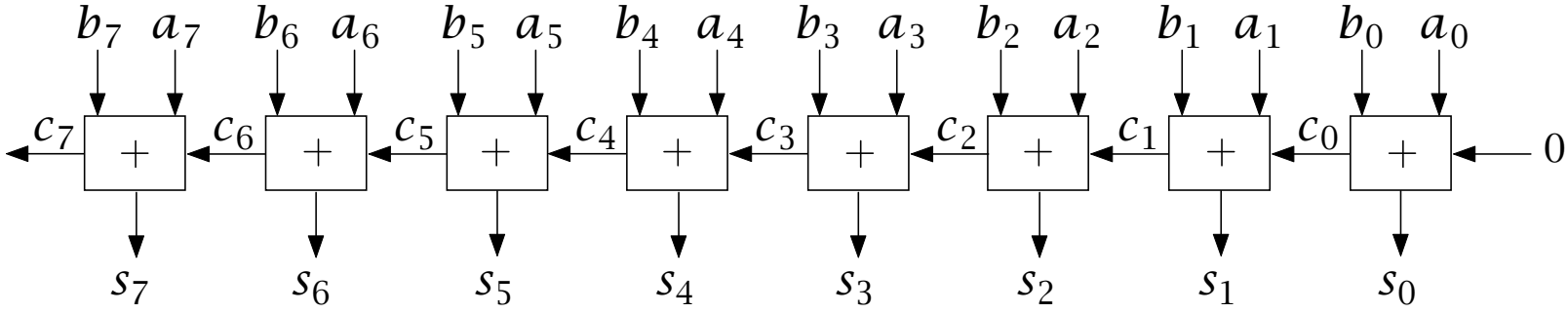
Connecting to the hardware

- We have set up a default .ucf file. Except for the clock all name assignments are commented out. Uncomment only those used for a given project.
- Use the names defined in the .ucf file as top's port signal names.
- When combining self contained modules that already have a top use a toptop to connect modules. For example an OFDM receiver with the XVGA support.

```
# light emitting diodes
#
NET "led<0>" LOC = "K12" | IOSTANDARD = LVCMOS33;
NET "led<1>" LOC = "P14" | IOSTANDARD = LVCMOS33;
#
# slide switches
#
NET "swt<0>" LOC = "F12" | IOSTANDARD = LVCMOS33;
NET "swt<1>" LOC = "G12" | IOSTANDARD = LVCMOS33;
NET "swt<2>" LOC = "H14" | IOSTANDARD = LVCMOS33;
```

Ripple carry adder

Having a one-bit adder we can use it to make an N-bit adder.

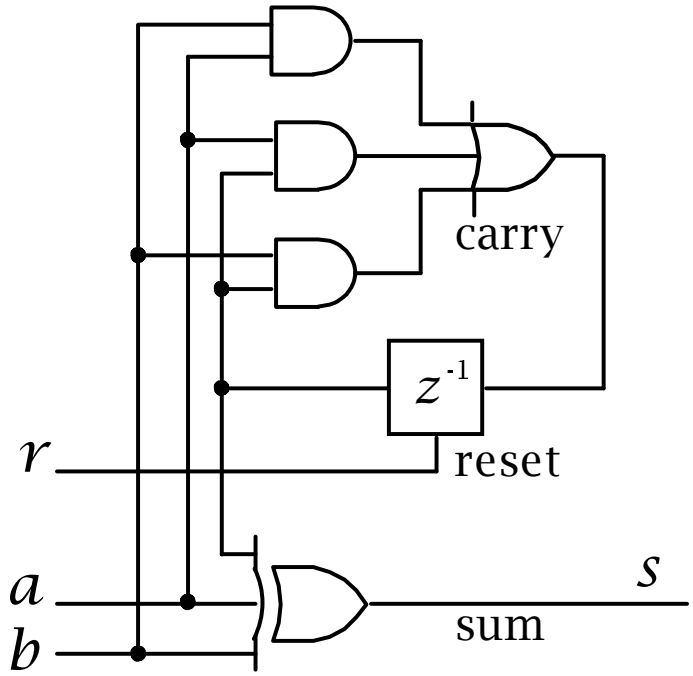


Execution time limited by the time required for carries to propagate from least significant bit to most significant bit..

Can reduce propagation time by incorporating carry lookahead logic. However this comes with a cost in terms of increased gate count. Knowing how to do this is where the pros start to earn their money.

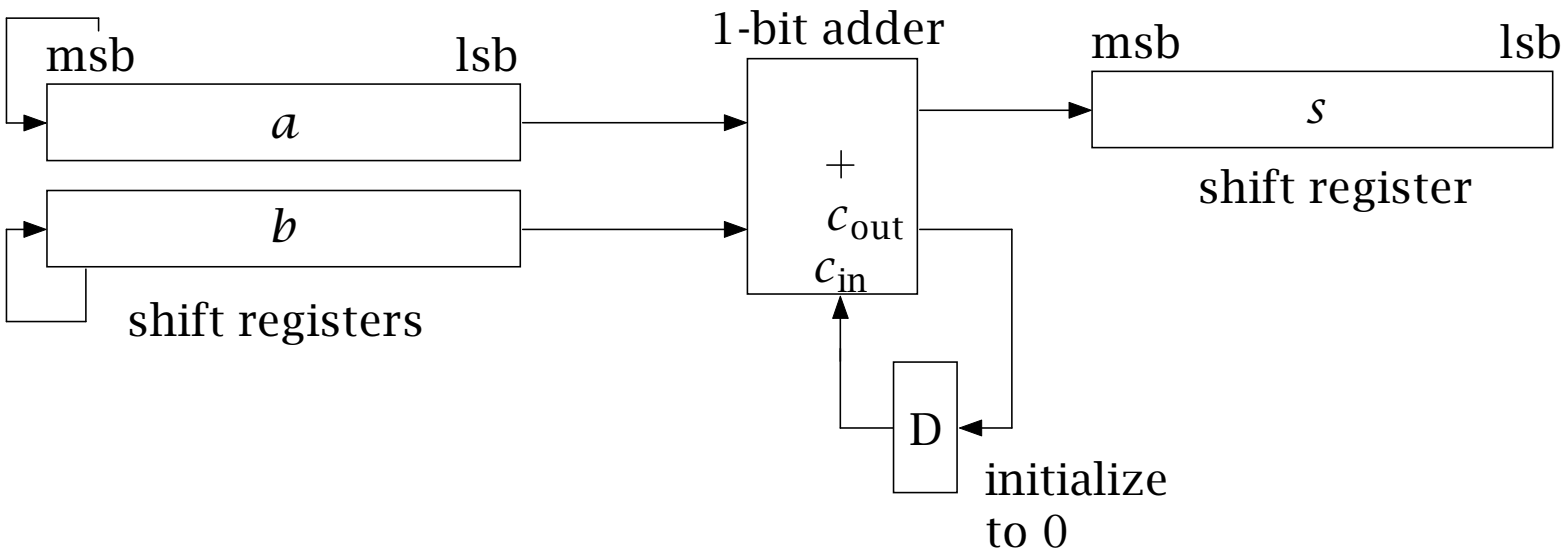
Bit serial addition

The basic element is a one-bit full adder with the carry bit fed back through a register. Values are shifted through the adder starting least significant bit first.



To subtract invert the bits of the value being subtracted and initialize the carry bit to one.

Bit serial adder



Minimal logic. Can be clocked at high rates.

Execution time strongly influenced by word size.

Not shown is the control logic needed to step the operation. This might be as simple as a counter. The point is that this extra logic is not shown.

VHDL for bit serial adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity BitSerialAdder is
    Port ( a : in std_logic;
           b : in std_logic;
           clear : in std_logic;
           clk : in std_logic;
           ce : in std_logic;
           sum : out std_logic);
end BitSerialAdder;

architecture Behavioral of BitSerialAdder is
    signal sum_internal:std_logic;
    signal carry_in:std_logic;
    signal carry:std_logic;
begin
    bit_sum:process(clear, clk) is  begin
        sum_internal <= a xor b xor carry_in;
        carry <= (a and carry_in) or (b and carry_in) or (a and b);
        if (clear = '1') then
            sum      <= '0';
            carry_in <= '0';
        elsif (rising_edge(clk)) then
            if (ce='1') then
                sum <= sum_internal;
                carry_in <= carry;
            end if;
        end if;
    end process bit_sum;
end Behavioral;
```

Comments

- Up to this point we have been using *combinational* logic.
- The bit serial adder introduces the use of *sequential* logic.
- VHDL's concept of sequential differs SIGNIFICANTLY from of that one thinks of as sequential when programming in a language such as C.
- Sequential statements have to be placed within a *process* block.

Becoming proficient

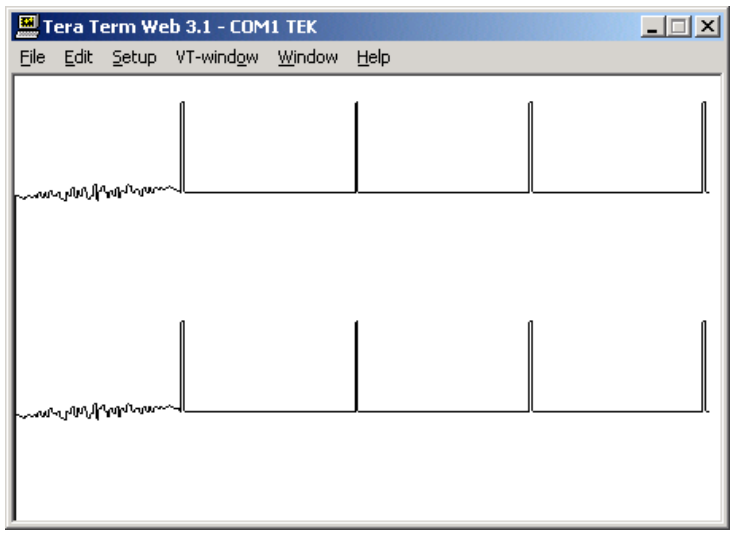
Sort of like playing the piano. You might

- be born meant to play it. It's in your body and soul.
- be good at it.
- be competent but not brilliant.
- have to work hard to get an audience.
- start looking for something else to do.

Becoming accomplished and successful requires desire, effort and practice, practice, ...

A small digression

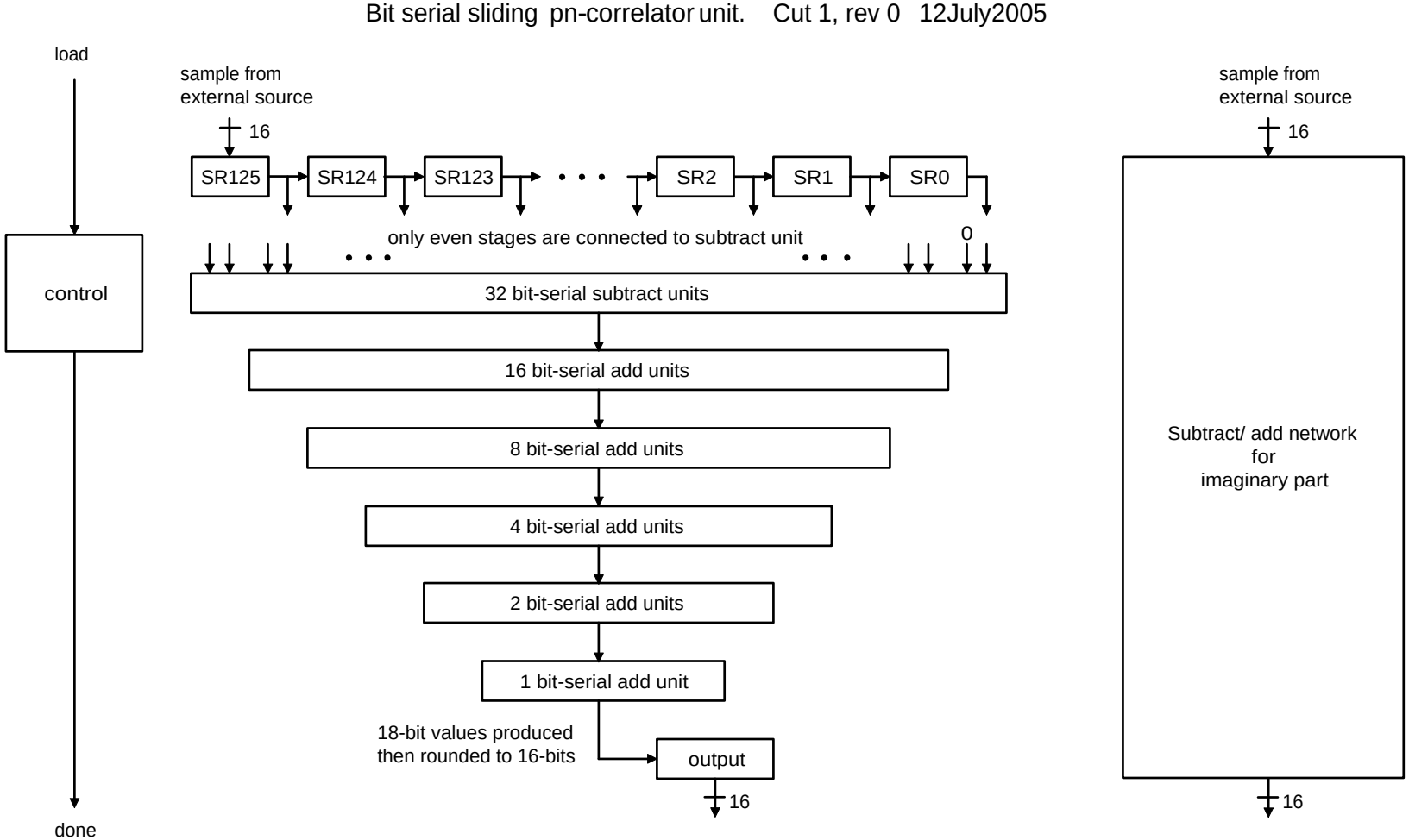
There exist families of binary sequences that possess periodic two-level autocorrelation functions. With proper processing the amount of energy contained in a full period can be collapsed into a single bit duration. An L-bit sequence has the potential to increase detection by $10\log(L)$ dB. These sequences are commonly used in communications systems.



Magnitude of the output of FPGA correlator via a PicoBlaze test processor. UART used to send formatted values to TeraTerm Tektronix 4100 emulator. Top plot is real part, bottom plot is imaginary part. The startup transient is seen at the left. Once a full period has been acquired the non-peak values are equal to -1.

Implementing a bit serial correlator

FPGA implementation adds/subtracts 63 complex values using a nominal 20 clock tics.



Another example project

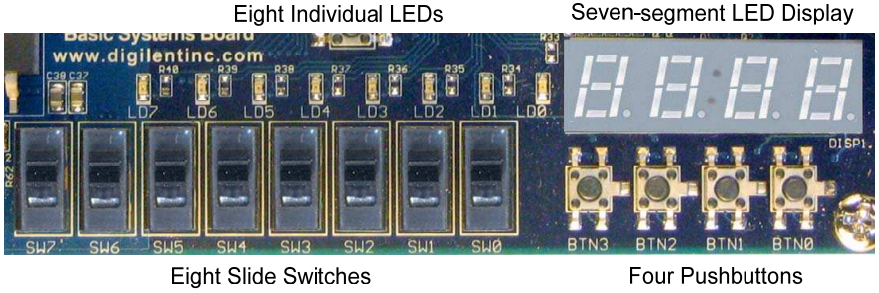
This is for the Basys. The only files that really needing to be changed for the S3-Starter Board should be the constraint file. Oops, also have to specify the correct FPGA part and package to ISE.

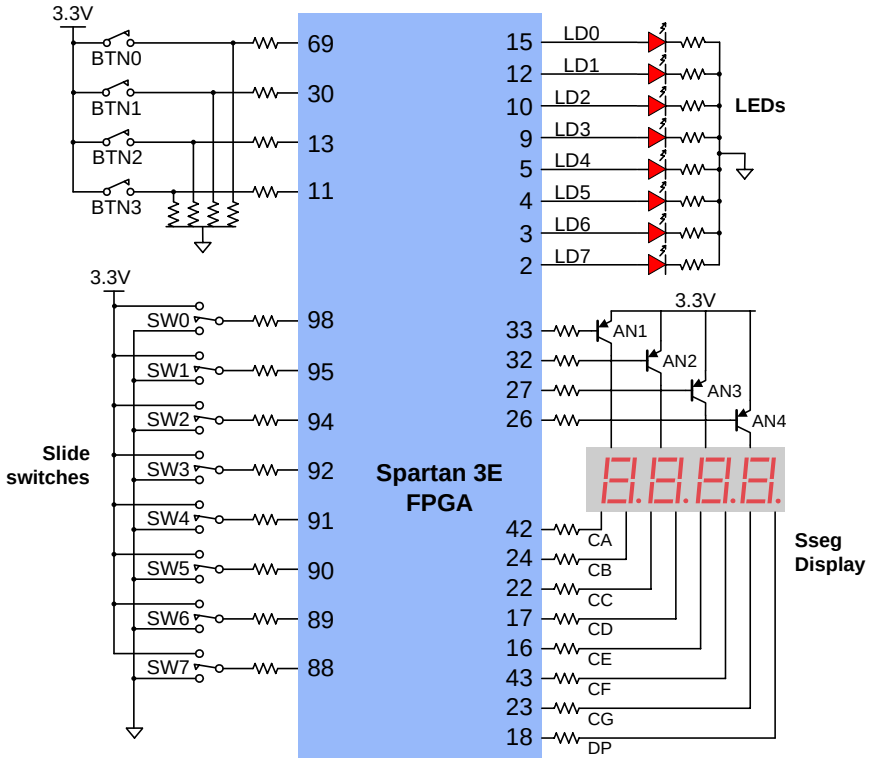
The task is to use the slide switches to turn on and off individual segments (and the dp) on the right most seven segment digit.

This is a purely combinational project and is intended to give practise using the tools.

Digilent's Basys Version C board used the 3E-100 TQ100 part while the current Version E board uses the 3E-100 TQ144 part.

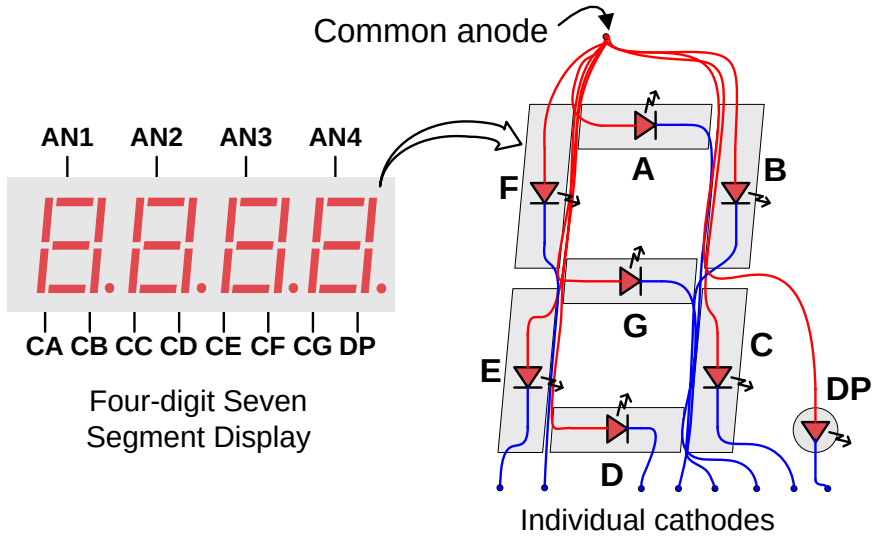
Basys board's devices…S3SB is similar

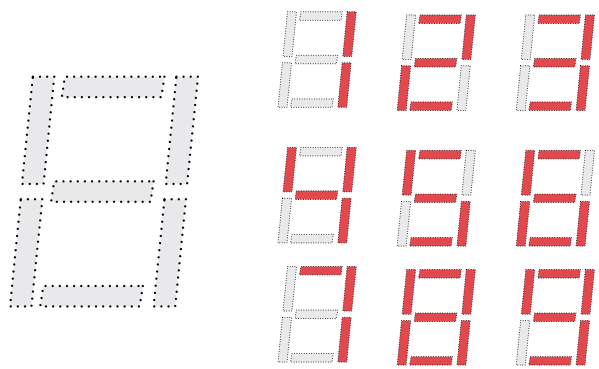

Eight Individual LEDs
Seven-segment LED Display
www.digilentinc.com
LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
BTN3 BTN2 BTN1 BTN0
Eight Slide Switches
Four Pushbuttons



From Digilent Basys Version C user manual.

Seven segment detail…S3SB similar





From Digilent Basys Version C user manual.

My simple 7-segment test VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity top_seven_segment is
    Port (  ssg : out  STD_LOGIC_VECTOR (7 downto 0);
            an  : out  STD_LOGIC_VECTOR (3 downto 0);
            swt : in  STD_LOGIC_VECTOR (7 downto 0));
end top_seven_segment;

architecture Behavioral of top_seven_segment is

signal sw_in : std_logic_vector(7 downto 0);
signal ssg_out : std_logic_vector(7 downto 0);

begin
    an <= "1110";
    sw_in <= not swt; -- complement switch bit pattern
    ssg_out <= sw_in; -- connect switches directly to segments
    ssg <=  ssg_out(7) & ssg_out(6) & ssg_out(5) & ssg_out(4) &
            ssg_out(3) & ssg_out(2) & ssg_out(1) & ssg_out(0);
end Behavioral;
```

My UCF

```
# seven segment digit anodes
NET "an<0>"  LOC = "P34" | IOSTANDARD = LVCMOS33; # left most digit
NET "an<1>"  LOC = "P33" | IOSTANDARD = LVCMOS33;
NET "an<2>"  LOC = "P32" | IOSTANDARD = LVCMOS33;
NET "an<3>"  LOC = "P26" | IOSTANDARD = LVCMOS33; # right most digit
# seven segment digit cathodes
NET "ssg<0>" LOC = "P83" | IOSTANDARD = LVCMOS33; # segment G
NET "ssg<1>" LOC = "P17" | IOSTANDARD = LVCMOS33; # segment F
NET "ssg<2>" LOC = "P20" | IOSTANDARD = LVCMOS33; # segment E
NET "ssg<3>" LOC = "P21" | IOSTANDARD = LVCMOS33; # segment D
NET "ssg<4>" LOC = "P23" | IOSTANDARD = LVCMOS33; # segment C
NET "ssg<5>" LOC = "P16" | IOSTANDARD = LVCMOS33; # segment B
NET "ssg<6>" LOC = "P25" | IOSTANDARD = LVCMOS33; # segment A
NET "ssg<7>" LOC = "P22" | IOSTANDARD = LVCMOS33; # decimal point
# slide switches
NET "swt<0>" LOC = "P38" | IOSTANDARD = LVCMOS33; # right most slide switch
NET "swt<1>" LOC = "P36" | IOSTANDARD = LVCMOS33;
NET "swt<2>" LOC = "P29" | IOSTANDARD = LVCMOS33;
NET "swt<3>" LOC = "P24" | IOSTANDARD = LVCMOS33;
NET "swt<4>" LOC = "P18" | IOSTANDARD = LVCMOS33;
NET "swt<5>" LOC = "P12" | IOSTANDARD = LVCMOS33;
NET "swt<6>" LOC = "P10" | IOSTANDARD = LVCMOS33;
NET "swt<7>" LOC = "P6"  | IOSTANDARD = LVCMOS33; # left most slide switch
```

Note: this is for the Basys-E board (current version)!

Xilinx ISE

- GUI entry system and design compiler.
- Supports graphical and text inputs.
- Does immense amount of work going from description to bit file.

Xilinx Impact

- Used to program FPGAs and PROMs.
- Use parallel-cable.
- Converts bit file to xxx.

Digilent Export

- Used to program FPGAs and PROMs.
- Part of Digilent's Adept software.
- Works with all boards via parallel-cable.
- ISE does not recognize Digilent USB cables.
- Really easy to use.
- Use bit file for the FPGA.
- Needs xxx file for the ROM.